PATENT ABSTRACTS OF JAPAN

(11)Publication number : **09-269931**
(43)Date of publication of application : **14.10.1997**

(51)Int.Cl. **G06F 15/00**
G06F 13/00
H04N 7/14

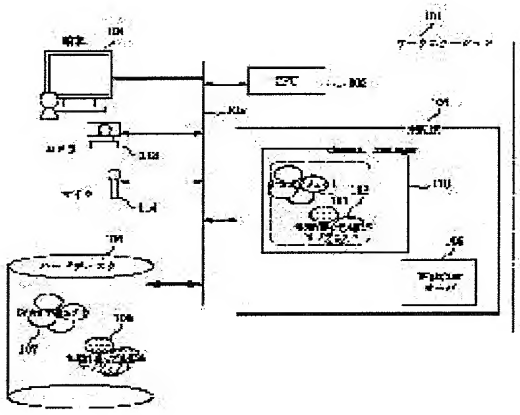
(21)Application number : **08-152326**
(22)Date of filing : **13.06.1996**

(71)Applicant : **CANON INC**
(72)Inventor : **FUKAZAWA TOSHIHIKO**

(30)Priority
Priority number : **08 14198** Priority date : **30.01.1996** Priority country : **JP**

(54) COOPERATIVE WORK ENVIRONMENT CONSTRUCTING SYSTEM, ITS METHOD AND MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To establish a cooperative work environment capable of coping with various forms of cooperative work and to execute an operation in it by providing a means for describing the cooperative work form and a means for processing cooperative work management adaptive to the optional cooperative work form based on described cooperative work form description.
SOLUTION: A hard disk 106 holds two kinds of data groups. One is a C/A object 107 and the other is a cooperative work model description object 108. These kinds of data are managed by a Watcher server or a server program called Watcher 109. A Session manager 110 transmits an animation and voice fetched from a camera 113 and a microphone 114 to another session manager, displays the animation received from another Session manager in a display 104 and reproduce voice so that conference between remote places is realized.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－269931

(43)公開日　平成９年(1997)10月14日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/00	３９０		Ｇ０６Ｆ	15/00	３９０
	13/00	３５１			13/00	３５１Ｇ
Ｈ０４Ｎ	7/14			Ｈ０４Ｎ	7/14	

審査請求　未請求　請求項の数29　ＯＬ　（全 24 頁）

(21)出願番号　特願平8－152326

(22)出願日　平成８年(1996)６月13日

(31)優先権主張番号　特願平8－14198

(32)優先日　平８(1996)１月30日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　深澤　寿彦
東京都大田区下丸子３丁目30番２号キヤノン株式会社内

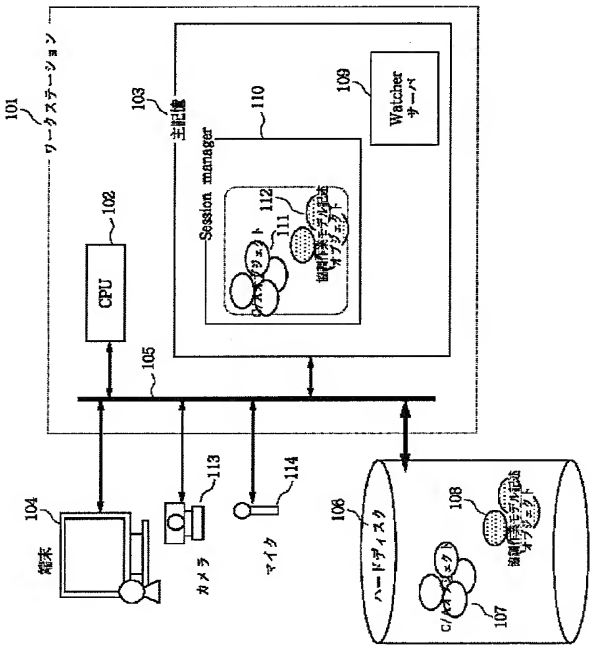
(74)代理人　弁理士　丸島　儀一

(54)【発明の名称】　協調作業環境構築システム、その方法及び媒体

(57)【要約】

【課題】　複数のコンピュータによって構成される作業環境において、さまざまな形態の協調作業に対応可能な協調作業環境実現方式および装置を提供する。

【解決手段】　複数のコンピュータによって構成される、協調作業をサポートする協調作業環境およびその実現機構であって、協調作業形態の記述手段と、上記記述手段によって記述された協調作業形態記述に基づき、任意の協調作業形態に合わせた協調作業管理処理をおこなう手段を備えた機構が開示される。



1

【特許請求の範囲】

【請求項１】　複数のコンピュータの、協調作業をサポートする協調作業環境を実現すべく、協調作業形態の記述手段と、上記記述手段によって記述された協調作業形態記述に基づき、任意の協調作業形態に合わせた協調作業管理処理をおこなう手段とを有することを特徴とする協調作業環境構築システム。

【請求項２】　請求項１において、上記協調作業形態の記述手段として、協調作業の参加者と参加者間の情報伝達路をあらわす概念を使用することを特徴とする、協調作業環境構築システム。

【請求項３】　請求項２において、上記協調作業形態の記述手段として、さらに上記協調作業の参加者と上記情報伝達路から構成される協調作業環境をあらわす概念を使用することを特徴とする協調作業環境構築システム。

【請求項４】　請求項３において、情報伝達が可能な上記情報伝達路の集合をもって、協調作業をあらわすことを特徴とする協調作業環境構築システム。

【請求項５】　請求項４において、上記協調作業形態の記述手段をクラスとオブジェクトとして提供することを特徴とする協調作業環境構築システム。

【請求項６】　請求項４において、上記協調作業形態の記述手段をスクリプトとして提供することを特徴とする協調作業環境構築システム。

【請求項７】　請求項１において、上記協調作業形態記述として、対象とする協調作業形態の構成要素と構成要素に対する操作を、上記協調作業形態の記述手段によって記述することを特徴とする協調作業環境構築システム。

【請求項８】　請求項７において、上記協調作業形態の構成要素と上記構成要素に対する操作をクラスとオブジェクトとして記述することを特徴とする協調作業環境構築システム。

【請求項９】　請求項７において、上記協調作業形態の構成要素と上記構成要素に対する操作をスクリプトとして記述することを特徴とする協調作業環境構築システム。

【請求項１０】　請求項７において、上記協調作業管理処理をおこなう手段として、上記協調作業形態の構成要素と上記構成要素に対する操作を記述したクラスとオブジェクトを使用することを特徴とする協調作業環境構築システム。

【請求項１１】　請求項７において、上記協調作業管理処理をおこなう手段として、上記協調作業形態の構成要素と上記構成要素に対する操作を記述したスクリプトを解釈するインタプリタを提供することを特徴とする協調作業環境構築システム。

【請求項１２】　複数のコンピュータの、協調作業をサポートする協調作業環境を実現すべく、協調作業形態の記述し、記述された協調作業形態記述に基づき、任意の

2

協調作業形態に合わせた協調作業管理処理をおこなうことを特徴とする協調作業環境構築方法。

【請求項１３】　請求項１２において、上記協調作業形態の記述方法として、協調作業の参加者と参加者間の情報伝達路をあらわす概念を使用することを特徴とする、協調作業環境構築方法。

【請求項１４】　請求項１３において、上記協調作業形態の記述方法として、さらに上記協調作業の参加者と上記情報伝達路から構成される協調作業環境をあらわす概念を使用することを特徴とする協調作業環境構築方法。

【請求項１５】　請求項１４において、情報伝達が可能な上記情報伝達路の集合をもって、協調作業をあらわすことを特徴とする協調作業環境構築方法。

【請求項１６】　請求項１５において、上記協調作業形態の記述方法をクラスとオブジェクトとして提供することを特徴とする協調作業環境構築方法。

【請求項１７】　請求項１５において、上記協調作業形態の記述方法をスクリプトとして提供することを特徴とする協調作業環境構築方法。

【請求項１８】　請求項１２において、上記協調作業形態記述として、対象とする協調作業形態の構成要素と構成要素に対する操作を、上記協調作業形態の記述方法によって記述することを特徴とする協調作業環境構築方法。

【請求項１９】　請求項１８において、上記協調作業形態の構成要素と上記構成要素に対する操作をクラスとオブジェクトとして記述することを特徴とする協調作業環境構築方法。

【請求項２０】　請求項１８において、上記協調作業形態の構成要素と上記構成要素に対する操作をスクリプトとして記述することを特徴とする協調作業環境構築方法。

【請求項２１】　請求項１８において、上記協調作業管理処理をおこなう手段として、上記協調作業形態の構成要素と上記構成要素に対する操作を記述したクラスとオブジェクトを使用することを特徴とする協調作業環境構築方法。

【請求項２２】　請求項１８において、上記協調作業管理処理をおこなう手段として、上記協調作業形態の構成要素と上記構成要素に対する操作を記述したスクリプトを解釈するインタプリタを提供することを特徴とする協調作業環境構築方法。

【請求項２３】　複数のコンピュータの協調作業動作間の移行の際に該協調作業の構成要素間の関連を保持し、前記移行を行うことを特徴とする協調作業システム。

【請求項２４】　前記協調作業は遠隔状況把握アプリケーションを含むことを特徴とする請求項２３の協調作業システム。

【請求項２５】　前記協調作業動作は会議アプリケーションを含むことを特徴とする請求項２３の協調作業シス

テム。

【請求項２６】　前記構成要素はソフトウェアのモジュールであることを特徴とする請求項２３の協調作業システム。

【請求項２７】　前記構成要素は前記ソフトウェアによって動作が制御されるハードウェア資源であることを特徴とする請求項２６の協調作業システム。

【請求項２８】　前記ハードウェアはテレビカメラであることを特徴とする請求項２７の協調作業システム。

【請求項２９】　請求項２３のシステムのための制御手順が格納された媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のユーザが複数のコンピュータを利用して協調作業、特にテレビ会議に好適な協調作業をおこなうことを可能にする協調作業環境の実現機構及びその方法、媒体に関するものである。

【０００２】特に、さまざまな協調作業形態に応じた協調作業環境の提供を可能にする、協調作業環境の実現機構に関するものである。

【０００３】

【従来の技術】近年、複数のパーソナルコンピュータあるいはワークステーションを組み合わせ、データ交換あるいはデータ共有を可能にすることにより、多人数での協調作業をサポートするグループウェア・システムが提案され、実際に利用されるされるようになってきている。

【０００４】このグループウェア・システムにおいて、コンピュータを介さずにおこなわれていた会議、プレゼンテーション、講演会などの協調作業をコンピュータ上でおこなうことを可能にする方法として、2種類のアプローチが存在する。

1．特定の協調作業(会議,講演会,etc.)を積極的にサポートする。

2．特定の協調作業形態へのサポートは特におこなわないかわりに、協調作業の場(たとえば会議室)の管理のみをおこなう。

【０００５】1.のアプローチによるものが、多くの商用テレビ会議システムなどであり、コンピュータ上の汎用協調作業システム(ShowMe(TM), Communique!(TM),など)の多くが2のアプローチを採用している。

【０００６】このように、1は特定の協調作業に特化したサービスが可能であり、2によったシステムは特定の協調作業をサポートせず、共通部分のみのサービス(すなわち、参加者と、使用されているアプリケーション・プログラムの情報の管理のみ)おこなうことにより、ユーザにとっては、さまざまな用途に利用することが可能である。

【０００７】

【発明が解決しようとする課題】しかしながら、従来のグループウェア・システムは1,2どちらのアプローチにおいてもそれぞれ問題点を持っている。

【０００８】1.のアプローチでは、当然のことながら特定の形態の協調作業(会議など)に特化しているため汎用性に乏しい。また、システムの拡張(新機能の追加など)も基礎となっている協調作業モデルを大きく逸脱することができないので、システムとしての発展性に欠けている。また、商用レベルのシステムにおいて、会議等の協調作業形態を十分にコンピュータ上などで実現しているとは言い難いものも存在する。

【０００９】2.ではシステム側では最低限のサポートしか提供しないため、実際の協調作業を支援する手段は、その上で動作するアプリケーション・プログラムが提供しなければならない。このため、アプリケーションの開発者の負担が増し、アプリケーション開発が困難となる。また、複数のアプリケーションを同時に使用している場合、各アプリケーションが想定している協調作業モデルが統一されていないと、ユーザを混乱させたり、アプリケーション間の連係に障害が発生する、という問題をもっている。

【００１０】本発明はかかる問題を個々に或いは全て解決することを目的とする。

【００１１】

【課題を解決するための手段】そこで、この発明では上記の問題を解決するために、協調作業をサポートする協調作業環境を実現すべく、

・協調作業形態の記述手段と、

・上記記述手段によって記述された協調作業形態記述に基づき、任意の協調作業形態に合わせた協調作業管理処理をおこなう手段を提供することにより、さまざまな形態の協調作業に対応可能な協調作業環境とその上で動作するアプリケーションの実現手段を提供するものである。

【００１２】

【発明の実施の形態】ここでは、本発明の第1の実施例について図を用いながら説明する。

【００１３】図1は、本実施例である会議システムの構成図である。図において、ワークステーション(101)は本実施例における処理の実行をおこなうCPU(102)と、本会議管理システムの基本ソフトウェアであるSession manager(110)を保持する主記憶(103)を計算機バス(105)で結合することにより構成されている。

【００１４】また、本実施例における処理に必要な永続データを保持するためのハードディスク(106)と、ユーザがシステムの操作をおこなうためのユーザインタフェースを提供するための端末(104)、ユーザの画像を撮るためのカメラ(113)および音声入力のためのマイクロフォン(114)がワークステーション(101)に結合されている。

【００１５】ハードディスク(106)は2種類のデータ群を保持している。1つはC/Aオブジェクト(107)であり、もう1つは協調作業モデル記述オブジェクト(108)である。これらのデータはWatcherサーバあるいはwatcher(109)とよばれるサーバ・プログラムにより管理されている。図1においてはハードディスク(106)およびwatcher(109)はワークステーション(101)の一部となっているが、通常はコンピュータ・ネットワークで通信可能な他のワークステーションに結合され、Session manager(110)などと同一マシン上で使用されることは稀である。

【００１６】C/Aオブジェクト(107)、協調作業モデル記述オブジェクト(108)の詳細については後述するが、どちらも、C++プログラム言語で作成されたオブジェクトとして扱われている。また、これらのデータのコピーがSession manager(110)内にそれぞれ存在している(111, 112)。Session manager(110)はこれらの情報をあつかうとき、ハードディスク(106)を参照するのではなく、自分自身のアドレス空間内に存在するコピー(111, 112)を参照する。

【００１７】(107)と(111)および(108)と(112)のデータの一貫性はwatcher(109)によって保証されるこれは、(111),(112)の値を変更した場合に変更内容をwatcher(109)に通知することによって実現される。watcher(109)は変更通知にしたがって(107),(108)を変更し、かつその変更内容を他のSession manager(異なるワークステーション上で動作している)に通知する。

【００１８】Session manager(110)は、カメラ(113)とマイク(114)から取り込んだ動画像および音声を他のSession managerに送信し、他のSession managerから受け取った動画像をディスプレイ(104)に表示し、音声を再生することにより遠隔地間の会議を実現する。

【００１９】図2はユーザの視点から見た、本実施例の概要をあらわす図である。図2において、複数のユーザ(201,202,203,204)がそれぞれコンピュータ・ネットワーク(205)で結合されたワークステーション上で作業をおこなっている、各ユーザのディスプレイ(206,207,208,209)には(210)および(211)の様なウィンドウが表示されており、ウィンドウ(210)においては協調作業の参加者をあらわすアイコン(212,213,214,215)が表示されている。

【００２０】本実施例では協調作業として、一般的な「会議」をサポートしている(この「会議」を本実施例ではセッションと呼ぶこともある。)。

【００２１】セッション内の参加者には、「議長」とよばれる役割(ロールと呼ぶ)が割り当てられることがある。図中で強調表示されているアイコン(212)は、そのユーザが「議長」であることを示している。セッションを招集した参加者が「議長」となるので、「議長」は1つのセッションに1人だけ存在することになる。「議長」の役割は発言者の指名をおこなうことである。

【００２２】ウィンドウ(211)は現在の発言者の動画イメージおよび音声を表示している。発言者のこれらのデータは全参加者のディスプレイに(206)上と同様にして表示される。発言者のみがセッションの全参加者に動画・音声情報を提供することができる。本実施例では、簡単のため発言者は一時に1人のみとする。「議長」の指名がなされるまでは「議長」は発言者でもある。発言者も「議長」と同様に、「発言者」というロールを割り当てられたユーザであるとみなすことができる。

【００２３】本実施例においてはまず、以上のような協調作業形態のもつ性質を定義・記述する手段を提供することを特徴とする。

【００２４】本実施例では記述手段として、C/Aオブジェクト・モデルというC++プログラム言語で記述されたC++のクラス群を提供する(このクラス群のことをC/Aクラスと呼ぶ。またC/AクラスのインスタンスをC/Aオブジェクトと呼ぶことにする。)。

【００２５】本実施例においては、協調作業モデルの記述手段としてC++によるC/AクラスおよびC/Aオブジェクトを採用するが、本発明はC++が提供する機能には限定されないので、同様な記述はObjective-C, C, CLOS, Lispなどさまざまなプログラム言語を用いても実現可能である。

【００２６】システムやアプリケーション等のプログラマはC/Aオブジェクトを利用して処理対象となる協調作業形態を記述する。

【００２７】C/Aクラスとしては以下のクラスが定義されている。

【００２８】

Attendee
Cpath
CollabField
Activity

これらのクラスのメソッドは、クラスがモデル化している実体の操作プリミティブに対応している。

【００２９】- Attendee
協調作業における参加者をモデル化したクラスである。

【００３０】- Cpath
ある参加者から他の参加者に対して形成される情報の伝達路をあらわすクラスである(CpathはCommunication pathの略である。)。C/Aオブジェクト上ではAttendee間の伝達路をあらわすことになる。情報の送信側のAttendeeのことを「送り手」、受信側のことを「受け手」とよぶ。

【００３１】- CollabField
図3にあるような、AttendeeとCpathが形成する協調作業の場ないし環境をあらわすクラスである(CollabFieldはCollaboration fieldの略である。)。図3において、(301)から(303)はAttendeeである。これらが協調作業の参加者に対応していると考えてよい。(304)から(308)はAt

tendee間の情報伝達路をあらわすCpathである。これらがCollabField(309)を形成している。

【００３２】図から分かるように、CpathはあるAttendeeから他のAttendeeへの一方向の伝達路であるが、必ずしも全てのAttendeeが対等にCpathを形成する訳ではない。図3においてAttendee(302)とAttendee(303)間のCpathは(305)だけである。これは、Attendee(303)から(302)への情報伝達ができないことをあらわしている。また、Attendee間に同じ方向のCpathを何本でも形成することができる。図3ではCpath(304)と(308)の2本の同方向のCpathが形成されている。これは、参加者間の情報伝達のために映像,音声,テキストなど複数の伝達手段を利用可能なことに対応している、ただし、1つのCpathが1つの伝達手段と対応しなければならない、という制約はない。1つのCpathが複数の伝達手段に対応することもありうる。

【００３３】- Activity

CollabFieldが形成され、Attendee間のCpathが形成されても、その上でかならず情報伝達がおこなわれるわけではない。このため、実際に情報伝達に使用されるCpathとそうでないCpathを区別する必要がある。この情報伝達に使用されるCpathのことをactiveなCpathとよび、使用されていないCpathをinactiveなCpathとよぶ。そして、CollabField上のactiveなCpathのみからなる集合のことをActivityと呼ぶ。CollabFieldが協調作業の場をあらわしているのに対し、Activityは協調作業の様子をあらわしている。図4は、図3で示したCollabField上にActivity(409)を重ねたものである。図において(401)から(409)までは図3の(301)から(309)までに対応している。ただし、activeなCpath(406,407)のみ実線で表記されており、inactiveな他のCpathは点線に書き直されている。図4におけるActivity(410)はCpath(406),(407)の＊み含んでいるので、現時点ではAttendee(401)とAttendee(403)の間のみで協調作業がおこなわれていることがわかる。

【００３４】先に挙げた「会議」のような協調作業形態は、C/Aオブジェクトの組み合わせにより記述することが可能である。この記述のことを特に「協調作業モデル記述」とよぶこともある。協調作業モデル記述は以下の部分から成り立っている。

【００３５】- 記述される協調作業の構成要素と、C/Aオブジェクトの対応関係

- 記述される協調作業に対して、参加者がおこなうことができる操作の宣言
- 上記操作に対応する処理の、C/Aオブジェクトの操作による記述

【００３６】本実施例における、セッション(「会議」)の協調作業モデル記述は以下のようになる。すなわち、まずセッションの構成要素には

- Session
- User
- Chairman
- Participant
- Speaker

が存在する。Sessionはセッションに対応する構成要素であり、1つの「会議」をあらわしている。Userはセッションの参加者のことである。またChairman, Speaker, Participantはそれぞれ「議長」,「発言者」,「その他の参加者」に対応するロールである。セッションの性質より、Session内にはかならず1人のロールChairmanをもったAttendeeが存在しなければならない。

【００３７】以上の構成要素とC/Aオブジェクトとの対応関係は以下のようになる。

【００３８】

Session	<-> CollabField
User	<-> Attendee
Chairman, Speaker, Participant	<-> 特定のAttendeeを送り手ないし受け手とするActivityの部分集合

【００３９】この対応関係は、上記の構成要素の挙動を、対応するC/Aオブジェクトの挙動として扱えることをあらわしている。

※【００４０】次に、「会議」に対してユーザがおこなうことができる操作を定義する。

※40 【００４１】

操作	意味
BeginSession	セッションの開始
TerminateSession	セッションの終了
AddUserToSession	セッションに参加者を加える
RemoveUserFromSession	セッションから参加者を削除する
BeSpeaker	「発言者」を指名する

ただし、以下の操作はその時点での「議長」のみが実行することができる。

【００４２】

【００４３】上記の操作は、C/Aオブジェクト上で以下のように引数と処理の記述をあたえることができる。(本来はC++を使用して記述されるが、説明の便宜上、自然言語をもちいて記述する。)。

【００４４】- BeginSession
引数: Userのリスト
1．与えられたUserのリストに対応するAttendeeのリストを作成する。
2．CollabFieldを2つ生成する。1方は「議長」を中心とする会議の管理のためのCollabFieldであり、他方は「発言者」を中心とする会議における情報伝達のためのCollabFieldである。(便宜上、前者をChairmanField、後者をSpeakerFieldとよぶことにする。)
3．この処理を要求したユーザを「議長」にすべく、BeChairmanを実行する。
4．この処理を要求したユーザを「発言者」にすべく、BeSpeakerを実行する。
5．2,3によって生成されたCollabFieldとActivityに対応させて、Sessionを生成する。(このCollabFieldおよびActivityは、生成されたSession内で管理される。)
【００４５】協調作業モデルの設計において、これらのCollabFieldをSessionに隠蔽せず、おのおのに対応する構成要素をわりあてることもできる。ただし、本実施例ではユーザにはこの2つの区別が自明ではないという立場に立ち、Sessionのみを定義した。また、セッションをChairmanFieldとSpeakerFieldに分割せずに、1つのCollabFieldでモデル化することも可能だが、CollabFieldの操作がより複雑になってしまう。
【００４６】BeginSessionによって生成されるCollabFieldおよびActivityは図5のようになる。図5において、2つのCollabFieldすなわちSpeakerField(501)とChairmanField(502)が存在する。まずSpeakerField(501)においては、3つのAttendee(503,504,505)が存在しそれぞれセッションの参加者に対応している。SpeakerField(501)上のすべてのAttendee(503,504,505)間には双方向に(506)から(511)までのCpathが生成される。
【００４７】ここで、強調表示されているAttendee(503)がロール「発言者」を保持しているとすると、Attendee(503)を送り手とし、残りのAttendee(504,505)を受け手とするCpath(507,510)のみがActivity(512)に属すことになる。inactiveなCpathが大部分をしめることになるが、これらのCpathも「発言者」が他のAttendeeに交替することにより、activeになることが可能である。
【００４８】また、ChairmanField(502)においてはSpeakerField(501)とまったく同じAttendee(503,504,505)が存在しているが、ここではロール「議長」を保持しているAttendee(504)を送り手とし、残りのAttendee(503,505)を受け手とするCpath(513,514)のみが形成されかつActivity(515)に属している。本システムにおいては、「議長」が他のAttendeeに移ることはないので(議長の退席は会議の終りとして扱われる)、SpeakerField(501)の様にinactiveなCpathを生成しておく必要はない。
【００４９】本実施例では、Cpathは無条件でactive/inactiveにすることができるが、これらに適当な条件を付加することにより、良い細かいデータ流の制御をおこなうことも可能である。
【００５０】- TerminateSession
引数: なし
1．Session内で管理されているCollabFieldとActivityを破壊し、存在していたすべてのCpathを消去する。
【００５１】－　ＡｄｄＵｓｅｒＴｏＳｅｓｓｉｏｎ
引数：　追加されるUser
1．追加されるUserに対応するAttendeeを生成する。
2．生成されたAttendeeをSession内の2つのCollabFieldに追加する。
3．2つのCollabFieldに、追加されたAttendeeと他のAttendeeを結ぶCpathを追加する。このとき、SpeakerFieldの場合は双方向の情報伝達ができる様に、双方向のCpathを形成する。また、ChairmanFieldでは「議長」になっているAttendeeから、追加されたAttendeeへのCpathが1つ追加されるだけである。
4．ChairmanFieldに追加されたCpathをactiveにし、ChairmanActivityに追加する。
5．SpeakerFieldに追加されたCpathのうち、「発言者」になっているAttendeeから追加されたAttendeeへのCpathをactiveにする。またそのCpathをSpeakerActivityに追加する。
【００５２】- RemoveUserFromSession
引数: 削除されるUser
1．削除されるUserに対応するAttendeeをとりだす。(UserとAttendeeの対応関係は、Userのデータ構造内にAttendeeが埋め込まれる形で管理されている。)
2．2つのCollabFieldから、削除されるAttendeeと他のAttendeeを結ぶすべてのCpathを削除する。
3．AttendeeをSession内の2つのCollabFieldから削除する。
【００５３】- BeSpeaker
引数: SpeakerになるUser
1．指定されたUserからAttendeeを取りだし、SpeakerField内のCpathでこのAttendeeを送り手としているものをすべてactiveにする。
【００５４】以上の様にして記述された協調作業形態の構成要素と操作は、C++のクラスとして実装されSession managerに組み込まれる。すなわち、各構成要素は1つのC++クラスとして実現され、各操作はそのC++クラスのメソッドとして実装される。本実施例の「会議」では、操作はSessionに関するもののみなので、上記の操作記述はSessionに相当するC++クラスのメソッドとなる。これらのクラスのインスタンスが協調作業モデル記述オブジェクトである。
【００５５】つぎに、組み込まれた協調作業モデル記述がシステムの挙動をコントロールする処理の流れを、図6の図を用いながら説明する。また図7でこの処理をフローチャートで表記したものを示す。図7は3つ短いフロー

チャート(7a,7b,7c)からなり、(7a)は図6におけるSession manager(605)の挙動をあらわし、(7b)と(7c)はそれぞれwatcher(612)とSession manager(606)の挙動を表記したものである。

【００５６】図6では、ネットワーク(613)で結合されたワークステーション(601,602,603,604)で3人のユーザ(608,609,610)が本システムを使用して会議をおこなっているものとする。各ユーザが使用しているワークステーション(602,603,604)では、Session manager(605,606,607)が起動され、ワークステーション(601)ではハードディスク(611)を管理するwatcher(612)が動作している。ハードディスク(611)内ではC/Aオブジェクト(614)と協調作業モデル記述オブジェクト(615)が管理されている。また、各Session manager(605)内ではC/Aオブジェクト群(614)のコピー(616)と協調作業モデル記述オブジェクト群(615)のコピー(617)が管理されている。

【００５７】さて図6において、ユーザ(608)がSession manager(605)の提供するユーザインタフェースを利用して、セッションの開始などの操作をおこなったと仮定する。

【００５８】Session manager(605)はユーザ(608)の操作を検知すると(S7a01)、協調作業システム記述オブジェクト(617)のメソッドを呼びだし、ユーザのおこなった操作をC/Aオブジェクト(616)上で実行する(S7a02)。本実施例の「会議」ではSessionに相当するC++クラスのユーザ操作に対応するメソッドが実行される。

【００５９】C/Aオブジェクト上の操作はwatcher(612)によってモニタリングされている(本実施例ではモニタリングの実現のために、C/Aオブジェクト上の操作には、かならず最後に、実行された操作をwatcher(612)に通知する処理が付加されているものとする。モニタリングは一部のデータベースが提供しているような、データの変更があったときに、その旨を通知する機構によっても実現可能である。)。

【００６０】watcher(612)は自身が管理しているC/Aオブジェクト上の操作を認識すると(S7b01)、そのオブジェクトの管理者に、操作がおこなわれたことを通知し、その操作に対応する処理の実行を要求する(S7b02)。

【００６１】ここで、オブジェクトの管理者とは、そのオブジェクトを生成したプロセスのことである(オブジェクト管理者ともいう。)。本実施例では、通常Session managerがC/Aオブジェクトを生成するので、Session managerがオブジェクトの管理者となる。

【００６２】ここでは、仮にSession manager(606)がオブジェクト管理者であったとすると、Session manager(606)が処理要求をうけとる(S7c01)、Session managerがうけとる処理要求は本会議システムにおいては、CpathのCollabFieldへの追加/削除および、Activityへの追加/削除である。(これは、先ほどの協調作業モデル記述の操作記述の内容から知ることができる。)。

【００６３】Session manager(606)はこれらの処理要求に応じて、以下の処理をおこなう(S7c02)。

【００６４】- CpathのCollabFieldへの追加
Cpathに対応する実際の動画・音声の伝達路を形成する。

【００６５】- CpathのCollabFieldからの削除
Cpathに対応する実際の動画・音声の伝達路を閉鎖する。

【００６６】- CpathのActivityへの追加
Cpathに対応する実際の動画・音声の伝達路に実際にデータを流す。

【００６７】- CpathのActivityからの削除
Cpathに対応する実際の動画・音声の伝達路でのデータ転送を停止させる。

【００６８】図7を見ると分かるように、Session manager(605,606)およびwatcher(612)の個々の動作は非常に単純である。協調作業モデル記述はこれらを組み合わせ方を記述しているとみなすこともできる。

【００６９】以上の様にして本実施例では、協調作業形態の記述手段としてのC/Aオブジェクトと、「会議」をC/Aオブジェクトによって記述した協調作業モデル記述を定義し、上記の定義にもとづいてシステムを制御するSession managerおよびWatcherサーバプログラムを提供した。

【００７０】さらに、議長と発言者をロールとしてモデル化した「会議」に対する協調作業モデル記述を提供し、ユーザインタフェースとしてSession managerプログラムを提供することにより、遠隔会議システムを実現している。

【００７１】本実施例では、協調作業形態として「会議」を記述しその実現について述べたが、本実施例のC/AオブジェクトおよびWatcherサーバは、「会議」のみではなくさまざまな協調作業形態を記述し、実装するために使用することが可能である。Sessin managerはユーザインタフェースを除き、協調作業形態に依存しないC/Aオブジェクト上の操作を処理するべく実装されている。このため、「会議」以外の協調作業システム上でSession managerを利用することが可能となる。

【００７２】本実施例では本発明を適用することにより、従来の会議システムよりも、役割のサポートなどの複雑な協調作業形態をサポートした遠隔会議システムを実現している。

【００７３】本実施例では発言者を1人に限定するなどの制限を課したが、これはC/Aオブジェクト上では、なくてもかまわない制限である。このため、C/Aオブジェクトでこれらの制限のない協調作業モデル記述を定義し、実現することが可能である。

【００７４】(その他の実施例) 第1の実施例において、会議管理システムにおける本発明の実施例についてのべたが、本発明の第2の実施例として、複数地点に配

置されたカメラを制御することができる遠隔状況把握システムにおける実施について説明する。

【００７５】この遠隔状況把握システムは、適当な位置に配置されたカメラの映像を適宜ディスプレイに呼びだし遠隔地の現在の状況を居ながらにして把握することを可能にするシステムである。会議などの厳密な意味での共同作業とは異なるが、共同作業の前段階としてのユーザ間の情報交換の手段を提供するという意味で、広義の共同作業システムとしてといえる。この遠隔状況把握システムも、本発明によって構築することが可能である。

【００７６】まず、図8を用いて本実施例の主要な構成要素について説明する。図8において、パーソナル・コンピュータ(以降単にマシンともよぶ)(801)は本実施例における処理の実行をおこなうCPU(802)と、本システムのソフトウェアであるVideoSender(810)とMultiViewer(811)を保持する主記憶(803)を計算機バス(805)で結合することにより構成されている。ただし、本システムはパーソナル・コンピュータにしか適用できないものではなく、実施例1と同様なワークステーション上でも実現することができる。VideoSender(801)はカメラ(809)の画像を取り込んでMultiViewer(811)に送信するプログラムである。MultiViewer(811)は受け取った画像をディスプレイ(804)上に表示するプログラムである。

【００７７】また、本システムが動作するために必要な共有データを保持するためのデータベース(806)と、ユーザがシステムの操作をおこなうためのユーザインタフェースを提供するディスプレイ(804)およびユーザの周辺の状況の画像を取り込むためのカメラ(809)がマシン(801)に結合されている。

【００７８】図1においては、VideoSender(810)とMultiViewer(811)が同一マシン上で動作しているが、本システムにおいてはどちらか一方がコンピュータ・ネットワークで結合された異なるマシン上で動作していてもよい。VideoSender(810)が他のマシンで動作する場合は、カメラ(809)もVideoSender(810)の存在するマシンに設置される。

【００７９】データベース(806)では、実施例1と同様な、ベースモデル情報(807)とセッション・モデル情報(808)が保持されている。これらは実施例1におけるC/Aオブジェクトおよび協調作業モデル記述オブジェクトと同じ情報を保持しているが、実施例1のようなC++オブジェクトとしてではなく、ascii文字列のテキストデータ形式となっている。

【００８０】データベース(806)そのものは、一般的なリレーショナル・データベースやオブジェクト指向データベースなど、データの永続化と一貫性管理の機構をもっているものであるが、本実施例ではこのようなデータベースで、データベース内のデータの更新がおこなわれた場合に、自動的にその旨をクライアントに通知する機構(変更通知機能)を持つものを採用する。このような機構を持たないデータベースを使用する場合は、実施例1のwatcherのようなサーバを別個に用意することにより変更通知機能を提供しなければならない。

【００８１】本実施例では、MultiViewer(811)とVideoSender(810)内のデータ形式と、データベース(806)内のデータ形式が異なる場合を想定している。このため、データ形式間の変換をおこなうモデルマッピングモジュール(812,813)がMultiViewer(811)とVideoSender(810)に組み込まれている。モデルマッピングモジュール(812,813)は、TCLプログラム言語のインタプリタとして実現されている。すなわち、データベース(806)内のベースモデル情報(807)とセッションモデル情報(808)はTCLで記述されたスクリプトおよび関連するデータが格納されている(TCLについては"Tcl and the Tk Toolkit",John K. Ousterhout, Addison-Wesley, 1994. 参照のこと)。モデルマッピングモジュール(812,813)は同時に、プログラム内の他のモジュールにセッションモデル情報(1115)やベースモデル情報(1114)の参照・更新および解釈実行のためのインタフェースを提供する。このため、プログラミングモデル的には、モデルマッピングモジュール(812,813)はC++におけるオブジェクトと同様な機能をはたす。

【００８２】本実施例では、データベース(806)上の情報のコピーは作成されない。必要に応じて、モデルマッピングモジュール(812,813)がデータベース(806)を参照する必要がある。

【００８３】つぎに、本システムをユーザの視点から見た概要について図9を用いて説明する。図9において、3人のユーザ(901,902,903)が個々のマシンで作業をおこなっている。各マシンはネットワーク(912)で結合されており、相互のデータ通信が可能である。

【００８４】ユーザ(901,902)にはカメラ(907,908)が設置されており、ディスプレイ(904,905)にはVideoSenderのユーザインタフェースであるウィンドウ(909,910)が表示されている。ユーザ(901,902)は、このウィンドウ(909,910)のユーザインタフェースを利用して、VideoSenderがカメラ(907,908)から取り込んだ画像のコントロール(大きさ，画質，他のユーザに画像を公開する／しないの指定)をおこなう。

【００８５】また、ユーザ(903)のディスプレイ(906)にはMultiViewerのユーザインタフェースを提供するウィンドウ(911)が表示されている。ウィンドウは、カメラ(907,908)からの画像を表示したり、別のマシンに設置されているカメラの画像を表示させるためのユーザインタフェースを提供している。

【００８６】図9においては、ユーザ(903)は他のユーザ(901,902)の画像を表示させ、状況把握をおこなうことができる。

【００８７】本実施例におけるベースモデル情報の記述においても、実施例1と同様に、

Attendee
Cpath
CollabField
Activity
を構成要素とする。その意味づけも同じである。ただ＊し、本実施例ではTCLを記述言語とするので、上記の構成要素はTCLの関数として定義される。これらの関数の基本的構造は以下のようになっている。

【００８８】

```
proc Attendee {args} {
        # 引数の先頭は操作プリミティブの指定
        eval "[lindex $args 0] $args";
}
```

【００８９】これは、上記のスクリプトはAttendeeを、引数の先頭に指定される操作プリミティブの名前をおなじ名前の関数を呼び出す手続きとして定義している。他の構成要素も同様にして定義される。

【００９０】さて、(遠隔)状況把握を実施例1と同様に協調作業モデル記述で表現すると以下のようになる。本実施例では、協調作業をベースモデルで記述したものをセッションモデル情報とよんでいるが、その作成の方法は実施例1とまったく同様にしておこなうことができる。すなわち、状況把握の構成要素を挙げると、

- AwarenessSession
- VideoSenderAttendee
- MultiViewerAttendee

となり、状況把握の活動を抽象化するAwarenessSessionと、ユーザ環境で起動される2つのプログラム、VideoSenderとMultiViewerのプロセスに対応するVideoSenderAttendee, MultiViewerAttendeeが定義される。

※

操作	意味
ConnectSender	VideoSenderとのコネクト
DisconnectSender	VideoSenderとのコネクト解除

が定義でき、VideoSenderAttendeeに対しては

操作	意味
MakeAvailableFor	特定のViewerへの映像の送信を許可する
MakeDisableFor	特定のViewerへの映像送信を禁止する

という操作が定義できる。

【００９５】これらの操作は、ベースモデル上で以下の様な処理としてあらわされる。本実施例でも以下の処理はTCLによって記述されるが、ここでは自然言語をもちいて説明する。

【００９６】- ConnectSender
引数: コネクト先のVideoSenderAttendee
1. 自分自身であるMultiViewerAttendeeが、まだAwarenessSessionを生成していないならば、AwarenessSessionを生成し、MultiViewerAttendeeに対応するAttendeeを、生成したAwarenessSession内のCollabFieldに追加する。
2. 与えられたVideoSenderAttendeeから対応するAttendeeを取りだし、AwarenessSession内のCollabFieldに追加する。
3. 追加されたAttendeeから自分自身をあらわすAttendeeへのCpathをAwarenessSession内のCollabFieldに追加する。

【００９７】- DisconnectSender
引数: コネクト先のVideoSenderAttendee
1. 与えられたVideoSenderAttendeeから対応するAttendeeを取りだし、AwarenessSession内のCollabFieldから削除する。このときそのAttendeeを受け手や送り手とするCpathはすべてCollabFieldから削除される。

【００９８】- MakeAvailableFor
引数: 映像を要求しているMultiViewerAttendee
1. 与えられたMultiViewerAttendeeから対応するAttendeeとAwarenessSessionを取りだす。
2. AwarenessSessionからさらにCollabFieldをとりだす。

※【００９１】VideoSenderAttendeeとMultiViewAttendeeはAttendeeに対応している。実施例1における「会議」ではユーザがAttendeeと対応づけられていたが、本実施例における「状況把握」ではプロセスがAttendeeと対応づけられており、ユーザとAttendeeの関連はモデル上は存在しないことになる。ただし、ユーザがプログラムを起動するのでまったく関連がないわけではない。

【００９２】AwarenessSessionは、CollabFieldと対応づけられるが、単なるCollabFieldの入れものとして定義され、ユーザに提供されるインタフェースからはAwarenessSessionのような概念は提示されない。強いていえば、図9におけるウィンドウ(911)がAwarenessSessionに対応している。

【００９３】次に、各サービスに対してユーザがおこなうことのできる操作は、MultiViewerAttendeeとVideoSenderAttendee各々に対して定義される。

【００９４】まず、MultiViewerAttendeeに対しては

3．取りだしたCollabFieldから自分自身であるAttendeeを送り手とし、MultiViewerAttendeeから取り出されたAttendeeを受け手とするCpathをactiveにする。
4．activeにしたCpathに対応する実際の動画・音声の伝達路への動画および音声の送信を開始する。
【００９９】- MakeDisableFor
引数: 映像を要求しているMultiViewerAttendee
1．与えられたMultiViewerAttendeeから対応するAttendeeとAwarenessSessionを取りだす。
2．AwarenessSessionからさらにCollabFieldをとりだす。

＊3．取りだしたCollabFieldから自分自身であるAttendeeを送り手とし、MultiViewerAttendeeから取り出されたAttendeeを受け手とするCpathをinactiveにする。
4．inactiveにしたCpathに対応する実際の動画・音声の伝達路への動画および音声の送信を停止する。
【０１００】以上の様にして記述された協調作業形態の構成要素と操作は、TCLの手続きとして記述されセッションモデル情報として、データベース内で管理される。たとえば、MultiViewerAttendeeは以下の様にして定義される。

＊　【０１０１】

```
proc MultiViewerAttendee {args} {
        global mv_attendee_table;
        # 引数の先頭は操作プリミティブの指定
        eval "[lindex $args 0]  $args";
}
```

【０１０２】上記のようなベースモデルにおける構成要素と同様な手続きに記述される。また、AttendeeとMultiViewerAttendeeの対応関係は、連想配列として定義されているmv_attendee_tableによって管理されている。この対応表もセッションモデル情報にふくまれる。尚、最後の式において”ｅｖａｌ”は評価を行うことを意味する。
【０１０３】このようにして記述された状況把握システムが形成するCollabFieldおよびActivityは図10のようになる。
【０１０４】すなわち、AwarenessSessionで形成されるCollabFieldは1つのMultiViewerAttendeeに対応するAttendeeと複数のVideoSenderAttendeeに対応するAttendeeから成り、後者から前者に向かうCpathのみが存在している。図10において、Attendee(1002,1003,1004)は各々1つのVideoSenderAttendeeに対応づけられており、MultiViewerAttendeeと対応づけられたAttendee(1001)を受け手とするCpath(1005,1006,1007)が形成されている。これらによって、CollabField(1008)が形成されている。
【０１０５】図においてはCpath(1005,1006)はActivity(1009)に属しているので、VideoSenderプロセスから実際に動画および音声データを受け取り表示することが可能だが、Cpath(1007)はinactiveなため、Attendee(1004)の管理者であるVideoSenderプロセスからはデータを受け取ることができない。ただし、状況次第でCpath(1007)はactiveになりうる。複数のMultiViewerに対応するAttendeeが存在する場合は、上記の様なCollabFieldおよびActivityを持つAwarenessSessionが複数存在する。ここでCpath(1007)はinactiveであったが、異なるAttendeeに対しては相当するCpathはactiveになっていてもよい。
【０１０６】次に、組み込まれた協調作業モデル記述がどの様にして、アプリケーションの挙動をコントロールするのかについて、図11を用いながら説明する。またここで説明する処理におけるモデルマッピングモジュール(1116)の動作をフローチャートであらわしたものを図12に示す。ここでは、モデルマッピングモジュール(1117,1118)の動作については特に説明しないが、その動作は図12でモデルマッピングモジュール(1116)がおこなうものとまったく同じである。
【０１０７】図11において、ネットワーク(1112)で結合されたマシン(1101,1102,1103)で３人のユーザ（１１０７，１１０８，１１０９）が本システムを使用して会議をおこなっているものとする。ユーザ(1108,1109)が使用しているマシン(1102,1103)にはカメラ(1110,1111)が設置されており、VideoSender(1104,1105)がそれぞれのマシンで起動されている。実際はユーザ(1108,1109)はさらにMultiViewerも起動することができるが、ここでは説明を簡単にするため、ユーザ(1107)の環境のみでMultiViewer(1106)が起動されているものとする。
【０１０８】さて図11において、ユーザ(1107)がMultiViewer(1106)に、VideoSender(1105)から動画を受け取り、ディスプレイに表示させることを要求したと仮定する(この時点で、MultiViewer(1106)とVideoSender(1105)間には、まだデータ転送のコネクションが形成されていないことが前提となっている。)。
【０１０９】MultiViewer(1106)はユーザからの要求を受け取ると、モデルマッピングモジュール(1118)のインタフェースを呼びだして、要求の処理を依頼する。モデルマッピングモジュール(1118)は上記の依頼をうけとると(S1201)、データベース(1113)から、セッションモデル情報(1115)とベースモデル情報(1114)をとりだし(S1202)、そこに書かれているTCLスクリプトを解釈して、ユーザからの要求を実行する(S1203)、この場合は、セッションモデル情報(1115)内に記述されている、MultiViewerAttendee手続きを実行することになる。この手続きは最終的にベースモデル情報上の操作として実行され、

その結果はモデルマッピングモジュール(1116)によってデータベース(1113)内のベースモデル情報(1114)およびセッションモデル情報(1115)に書き戻される(S1204)。

【０１１０】すると、データベース(1113)の変更通知機能によって、その結果はVideoSender(1105)に通知される。

【０１１１】VideoSenderは自身が管理しているベースモデル情報上の操作を認識すると、その操作に対応する処理を実行する。

【０１１２】VideoSender(1105)が、MultiViewer(1105)がおこなう処理の結果として受け取る処理要求は第1の実施例とおなじく以下のものである。

【０１１３】- CpathのCollabFieldへの追加
- CpathのCollabFieldからの削除
- CpathのActivityへの追加
- CpathのActivityからの削除

【０１１４】VideoSender(1105)は上記の処理要求に対して、以下の処理をおこなう。

【０１１５】- CpathのCollabFieldへの追加
Cpathに対応する実際の動画・音声の伝達路を形成する。

【０１１６】- CpathのCollabFieldからの削除
Cpathに対応する実際の動画・音声の伝達路を閉鎖する。

【０１１７】- CpathのActivityへの追加
まず、VideoSenderのユーザにMultiViewerからの状況参照が要求されている旨を表示し、ユーザの判断を仰ぐ。ユーザは、状況に応じてMakeAvailableForないし、MakeDisableForを実行する。MakeAvailableForが実行されると、Cpathはactiveなまま、Cpathに対応する実際の動画・音声の伝達路へのデータ送信が開始される。また、MakeDisableForが実行されるとCpathはinactiveになり、実際の動画・音声の伝達路への動画および音声の送信は停止される。

【０１１８】- CpathのActivityからの削除
Cpathに対応する実際の動画・音声の伝達路でのデータ転送を停止する。

【０１１９】以上の様にして本実施例では、協調作業形態の記述手段としてのベースモデルと、状況把握をベースモデルによって記述したセッションモデル記述を定義し、上記の定義を実行するモデルマッピングモジュールを提供した。

【０１２０】さらに、状況把握システム用のセッション・モデル記述を提供し、モデル・マッピングモジュールをVideoSender, MultiViewerという2つのアプリケーションに組み込むことにより、アプリケーションの協調作業モデルに即した制御をおこなう手段を提供した。

【０１２１】本実施例では、実施例1と異なる構成をとっているが、ベースモデルに実施例1と等価なものを使用しているので、実施例1の会議システムを本実施例上で実現することが可能である。この場合、「会議」も「状況把握」もベースモデルで記述できるので、「状況把握」から「会議」に移行する処理をベースモデルに追加することにより、この2つの協調作業形態を統合し相互に移行可能なシステムを実現することが可能である。

【０１２２】本実施例では本発明を適用することにより、ベースモデルおよびセッションモデルを、文字列によるスクリプトとして管理するため、ユーザはデータベースの機能を利用して適宜、その内容を変更することが可能である。このため、セッションモデルの変更・拡張を容易におこなうことができる。

【０１２３】また、ここで実現された状況把握システムそのものも、ベースモデルに基づいた構造を持っているので拡張性が高く、VideoSenderやMultiViewerの個数や構成に影響されずに動作することができる。VideoSenderとMultiViewerの挙動は最終的にベースモデル上の操作に対する処理として記述されているので、セッションモデルの拡張がおこなわれた場合も、VideoSenderやMultiViewerの変更はユーザインタフェースだけに限定できる。

【０１２４】次に本発明の他の実施例について説明する。かかる実施例においては、ＣＳＣＷ（Ｃｏｍｐｕｔｅｒ　Ｓａｐｐｏｒｔｅｄ　Ｃｏｏｐｅｒａｔｉｖｅ　ｗａｋ）のための空間的に分散した環境間で同期的に行われる協調作業を支援することを目的とした同期型のＣＳＣＷアプリケーション（以下ＣＳＣＷアプリと称す）の連携が説明される。

【０１２５】ここでは、遠隔地に設置されたビデオカメラの画像を参照するための「遠隔状況把握アプリ」と「ビデオ会議アプリ」の組み合わせにおいて、「状況把握アプリ」でユーザの在室を確認してから、そのユーザと「ビデオ会議アプリ」で作業を開始する、という状況を考える(図１３)。

【０１２６】この組み合わせにおいては図１３中の、「状況把握アプリ」２０００でのユーザの検索／確認作業から「ビデオ会議アプリ」３０００での共同作業へのスムーズな移行をサポートすることが望まれる。この要望を実現するためには、2つのCSCWアプリ間でのユーザ等に関する情報の受渡しや、状況把握アプリが使用していたカメラ等２００２の資源を効率良くビデオ会議アプリに受渡すなどの相互作用を行なって、ユーザインタフェース上の不整合や各CSCWアプリ間の資源の競合などをうまく解消しなければならない。同様な要求は、「状況把握アプリ」２０００と「ビデオ会議アプリ」３０００が同時に動作(共存)する場合においても発生する。

【０１２７】そこで「CSCWアプリの連携」を、上記の例の様な
・あるCSCWアプリから別のCSCWアプリへの移行
・2つ以上のCSCWアプリの共存
においてCSCWアプリ間に発生するさまざまな相互作用の

ことであると定義する。

【０１２８】CSCWアプリの連携は、連携の対象によって大きく以下の2種類に分けることが可能である(図１４)。

１．協調プロセス群間の連携２０１０

２．協調作業モデル間の連携２０２０

一般に、CSCWアプリは複数のホストに分散して動作するプロセス群として実現される。このため、CSCWアプリの移行／共存は、各CSCWアプリを構成するプロセス群(協調プロセス群と呼ぶ)の間の移行／共存としての側面を持っている。

【０１２９】「１．協調プロセス群間の連携」２０１０とは、この協調プロセス群間の移行／共存として実現する際に必要となる連携のことである。この連携は、CSCWアプリをいかに効率よく協調させるかという問題に関わっており、具体的には以下の2種類の連携が考えられる。

【０１３０】・共有可能なハードウェア資源およびソフトウェア・モジュールの重複の解消

移行の対象となるCSCWアプリ間で、同一のビデオカメラ,マイク等の資源を利用したり、同一のプログラム,プロセスを構成要素としている場合がある。この場合は、資源やプロセスの再利用や共有を可能にすることにより、移行／共存を効率良く実現することが可能となる。このサポートがなければ、同一プログラムを要求のたびに起動してしまう様な状況が発生してしまう等の問題が発生する。

【０１３１】・共有不能なハードウェア資源のアクセス制御

協調作業形態の共存においては、資源そのものが排他的なアクセスしか許さない様な状況も発生する。この場合、CSCWアプリ間での資源へのアクセス制御の機構が不可欠となる。

【０１３２】また、もう一つのタイプの連携である「２．協調作業モデル間の連携」２０２０とは、CSCWアプリがユーザに対して提供する「会議」,「講演会」,「共同執筆」などの協調作業モデル間の連携である。

【０１３３】この連携は、ユーザに対して複数のCSCWアプリを如何に統一的に提示するかというユーザインタフェースが重要である。その意味で協調作業モデル間の連携は、対象となる協調作業モデルによって様々な相互作用が発生する。代表的な連携として以下のものが挙げておく。

【０１３４】・協調作業モデルの構成要素間の関連の制御

協調作業形態が異なれば、その構成要素も異なってくる。しかし、移行／共存時には、その構成要素間に何らかの関連が発生することがある。

【０１３５】例えば、前述の様に「遠隔状況把握アプリ」２０００から「会議アプリ」３０００への移行を考える。ここで、「遠隔状況把握アプリ」２０００では、遠隔地に設置されたビデオ・カメラを協調作業形態の基本的な構成要素となる。これに対し、「会議アプリ」３０００では会議の参加者であるユーザを基本的な構成要素とする。

【０１３６】2つのCSCWアプリ間の移行において、移行前のビデオ・カメラに写っていた人物が移行後のユーザになるという関連が存在し得る。本実施例ではこの関連を制御可能にすることにより、カメラに写っていたはずのユーザを、「会議アプリ」３０００起動後に改めて会議の参加者として指定しなければならない、といった状況を避けることが可能となる。

【０１３７】以上のような連携をサポートする上で障害となるのは、移行／共存の対象となる各CSCWアプリにおける

・プロセス構成と資源へのアクセス形態の違い

・協調作業モデルの構成要素、およびその振舞いの違いが必ずしも明らかではないことである。これらの差異が明確でなければ、各構成要素の対応関係等を決定することができず、連携の記述(プログラミング)／制御の実現が困難になる。この問題はCSCWアプリの多様化に伴い、ますます「協調プロセス群間の連携」,「協調作業モデル間の連携」を実現する上での大きな障害となると考えられる。

【０１３８】したがって本実施例のグループウェア・フレームワークCCF(Collaboration Control Facility)では上記の問題に対し、協調プロセス群／協調作業モデルの両面について、

・その構成／形態／振舞いの記述を可能にし、

・その記述に基づいた、連携のプログラミングおよび制御手段を提供する、ことによりCSCWアプリの連携の実現をサポートする。

【０１３９】さて次いでCCFの概要について説明してから、協調プロセス群間の連携／協調作業モデル間の連携の各々についてのCCFにおける実現について説明していく。

【０１４０】グループウェア・フレームワークとしてのCCFは、C++ツールキットCCF Toolkitを提供する。CCF Toolkitは、協調プロセス群と協調作業モデルの構成／振舞い等を記述するためのプログラマ・インタフェースを提供する。また、CSCWアプリ間で行なわれる連携を記述するためのプログラマ・インタフェースもCCF Toolkitによって提供される。

【０１４１】CCF Toolkitを組み込んで作成されたCSCWアプリはCCF Core System ３５０１によって制御／管理される。

【０１４２】CCF Core System3501は、CCF Core ServerとCCF Core Interfaceという2種類のプログラムで構成されている。CCF Core ServerはCSCWアプリに関する情報を管理する集中サーバである。また、CCF Core Inter

faceは利用者に起動等CSCWアプリの制御インタフェースを提供するプログラムである。

【０１４３】図１５にCCF Core Systemの概念図を示す。

【０１４４】CCF Core Systemの内部は、2章で述べた2種類の連携のタイプに対応した以下の2つのレイヤから構成されている。

【０１４５】・Node modelレイヤ３５００

CSCWアプリがどの様なプロセスから構成され、各プロセスが構成要素としてどの様に関連しているかを管理するレイヤである。後述するNodeモデルというCSCWアプリのモデルに基づき、CSCWアプリのプロセスの起動,終了等の制御を行なう。連携の制御においては、協調プロセス群の間の連携機構を提供する。

【０１４６】・WorkEnvレイヤ３６００

CSCWアプリの協調作業モデルを扱うレイヤである。各CSCWアプリが提供する協調作業モデル上の操作と、Node modelレイヤでのプロセス単位の処理の対応を管理／制御する。連携の制御においては、協調作業モデル間の連携機構を提供する。

【０１４７】では、次に協調プロセス群および協調作業モデルにおける連携サポートについて説明する。

【０１４８】ここでは、CCFによる協調プロセス群の連携サポートについて説明する。協調プロセス群の連携は図１６に示すNode modelレイヤの管理下に置かれている。

【０１４９】Node modelレイヤは、

・CSCWアプリを構成する協調プロセス群を記述するためのNodeモデル

・Nodeモデルに基づく、連携の制御機構

を提供することにより連携をサポートする。

【０１５０】図１６に示すNodeモデルは、我々が検討しているプログラム／プロセス単位でのCSCWアプリ制御モデルである。このNodeモデルは以下の構成要素から成り立っている。

【０１５１】・Node4010、4020

CSCWアプリのプログラムが動作するユーザ環境やホストを抽象化したものである。後述するResourceやModuleの管理単位として機能する。

【０１５２】・Module4030

ModuleはCSCWアプリを構成するプロセスを抽象化したものである。Node内で管理されているModuleは、Nodeに対応するホスト上のプロセスとみなすことができる。

【０１５３】・Resource4050

Resourceはカメラ、マイク等の外部入出力機器を代表とする資源を表現する。Node内で管理されているResourceは、Nodeに対応するホストに設置されている外部入出力機器あるいはその管理サーバとみなすことができる。

【０１５４】・Channel4060

ChannelはModule4030およびResource4050間の通信路を表現する。1:1の通信だけでなく、broadcastの様な1:Nの通信を1つのChannelで表現することができる。また、Channelは同一Node内のModule,Resource間だけでなく、異なるNodeのModule,Resource間でも形成することができる。

【０１５５】Nodeモデルは、CSCWアプリの協調プロセス群が動作する各ホストにおける、

・動作するプロセス

・使用される資源

・プロセス間の通信路および資源へのアクセス・パス

を記述する。

【０１５６】Nodeモデルにより、CSCWアプリは、関連するホストと対応づけられたNodeの集まりとして表現される。上記の各構成要素には、それぞれ一意な名前が付加され、CCF Core Serverによって集中管理される。また、CCF Core ServerはNodeモデルを使用して、実際のCSCWアプリの振舞いをモニタし、必要に応じて制御を行なう。

【０１５７】上記の様なNodeモデルを形成するためには、個々のグループウェア・アプリがどの様なNodeを形成するかをあらかじめ記述しておく必要がある。

【０１５８】このNodeモデルの設計図となる情報をNodeフレーム情報と呼び、Nodeフレーム情報作成のためのインタフェースはCCF ToolkitのC++クラス群によって提供される。

【０１５９】NodeフレームもCCF Core Serverによって管理され、必要に応じてNodeモデルの作成に使用される。

【０１６０】前述したように、協調プロセス群の連携はCSCWアプリの移行／共存時に必要となるプロセスおよび資源の共有／アクセス制御として扱うことができる。

【０１６１】このプロセスおよび資源の共有／アクセス制御は、Node,Module,Resource,Channelの置換処理として記述／制御することが可能である。

【０１６２】図１７に、このNodeモデルでの連携処理のアルゴリズムを示す。図中で、2つのCSCWアプリAからBへの移行が行なわれるものとする。さらに、これらのアプリが両方ともNodeを形成するホストに注目すると、CSCWアプリAのNode4010からCSCWアプリBのNode4020への移行が行なわれるものとする(図１７－移行前)。このNode間の移行は、Node1のModule,Resource,Channelを用いてNode2を再構成することによって実現される。

【０１６３】この再構成処理は、移行前のNode1と移行後のNode2とにおいて共通のModule（図中の楕円）Resource（図中の矩形）が使用される場合には、これらを再利用する(図１７－移行中)。ただし、同一名であってもModule,Resourceに対して再利用が不可である旨が指定されている場合を除く。また、Channelについては同一名であっても、Channelの両端のModule,Resourceが再利用されていなければ、Channelは再利用されない。この

様して再利用されたModule,Resource,Channelでは、対応するプロセス等がCSCWアプリBの一部として使用される。

【０１６４】再利用される構成要素が決定されれば、Node2の生成に不足しているModule,Resourceを生成し、最後に全てのChannelを生成する(図１７－移行後)。この処理によって新たなプロセスが起動され、さらに再利用されたプロセスと新しいプロセス間での通信路が形成されることになる。

【０１６５】以上の処理をすべてのNodeに対して実行することにより、CSCWアプリAからCSCWアプリBへの移行が完了する。

【０１６６】また、CSCWアプリ間の共存の処理も上記の様にして行なわれるが、Resourceに対しては、どちらか一方のアプリしか資源にアクセスできない様な状況が発生する。この場合、Nodeモデル上では、両方のNodeに同じResourceが存在していることになる。ただし、そのResourceと他のModule間にChannelを形成できるのは一方のNodeだけである。必要に応じて、Channelを張り替えることにより、資源を共有するCSCWアプリ間でのアクセス制御をNodeモデル上で扱うことが可能となる。

【０１６７】以上の様に、CSCWアプリの開発者は、提供するCSCWアプリのNodeモデルによる記述と、その中で再利用可能なModule,Resource,Channelを記述することにより、他のCSCWアプリとの協調プロセス群間の連携に必要な情報を提供することができる。

【０１６８】Nodeモデルで記述されたCSCWアプリに対して、CCF Core SystemのNode modelレイヤはNodeモデル上の各構成要素と実際のホスト,プロセス,資源,通信路の対応関係を管理する。

【０１６９】そして、Nodeモデル上の操作を実際のホスト,プロセス,資源,通信路に反映させる。また逆に、実際のホスト,プロセス,資源,通信路の振舞いをモニタすることにより、結果をNodeモデル側に反映させるなどの処理を行なう。

【０１７０】Nodeモデルから実際への反映は、ユーザがCCF Core Interfaceなどを介してCSCWアプリの操作を要求したときに行なわれる。

【０１７１】実際のホスト,プロセス,資源,通信路からNodeモデルへの反映は、CSCWアプリ側が独自にアクションを起こしてから、その旨をCCF側に通知する場合に行なわれる。また、エラー等によるプロセスを異常終了をCCF Core Serverが検知した場合も、関連するModule,Resource,Channelの消去が行なわれる。

【０１７２】以下に、Nodeモデル上で可能な操作と、それに対応するCCF側の処理を列挙する。

・Nodeの生成および消去
Node内部のModule,Resource,Chanellの生成削除
・Moduleの生成および消去
対応するプログラムの起動／終了
・Resourceの生成および消去
対応する資源管理サーバの起動／終了
・Channelの生成および消去
対応する通信路の生成／消去処理

【０１７３】協調プロセス群の連携においては、アクセス制御等の実現のためにChannelの制御の実現が重要である。Channelに対応する通信路の制御は、CCF側からグループウェア・アプリのプログラムに対して送られる以下のメッセージのハンドラ手続きによって実行される。このメッセージ・ハンドラは、CSCWアプリの開発者によって提供され、socket,rpc等による実際の通信路の形成処理を行なう。

・CREATE-CHANNEL # Channelの生成
・DESTROY-CHANNEL # Channelの消去

通信路の形成処理は、Channelの両端のModuleないしResourceのどちらか1方がオーナ（図１８ではNode#1）となり、上記のメッセージを受け取って通信路の生成／消去を行なう(図１８)。また、オーナではないModule,Resourceに対応するプロセスには、通信路生成／消去の通知メッセージが、オーナ側の処理の終了後に送られる。通知メッセージには、socket,RPC等で通信路を確立するために必要なポート番号等の情報が付加されるので、通知メッセージを受け取ったプロセスが通信路を確立することが可能となる。

【０１７４】CCFのWorkEnvレイヤでは、協調作業モデルの連携における協調作業モデル間の情報の対応をあつかう。

【０１７５】Node modelレイヤにおける協調プロセス群の連携サポートの機構は、CCF CoreServerによって提供されたが、協調作業モデル間の連携制御機構はCSCWアプリ側によって提供される。

【０１７６】そのかわりに、CCFは

・各CSCWアプリが提供する「会議」,「講演会」といった協調作業モデルを記述するためのC++インタフェースと、
・2つの協調作業モデル間におけるユーザ等の構成要素の対応関係を記述／制御するためのC++インタフェースを提供する。

【０１７７】この協調作業モデルの記述は、

・協調作業モデルの構成要素と、Nodeモデルの構成要素の対応関係
・協調作業モデルの構成要素に対する操作と、Nodeモデル上の操作の対応関係

をプログラミングし、CCF Core Systemにそのプログラムを追加することによって行なわれる。このプログラムをWorkEnvManagerと呼んでいる。

【０１７８】また、協調作業モデル間の対応関係も同様にプログラムとして記述される。このプログラムはWorkEnvMoverと呼ばれる。WorkEnvMoverは、CSCWアプリ間の連携時にCCF Core ServerのWorkEnvレイヤによって起動

され、協調作業モデル間の情報の共有／再利用処理を実行する(図１９)。

【０１７９】WorkEnvManagerおよびWorkEnvMoverはCSCWアプリの一部としてアプリ側の開発者によって提供されることになるが、CCFはこれらのプログラムを利用して協調作業モデル間の連携を実現する。

【０１８０】本実施例ではCSCWアプリの連携は、各CSCWアプリを構成するプロセス群の間の相互作用と、CSCWアプリがユーザに提供する協調作業モデル間の相互作用としてあらわされる。

【０１８１】連携のサポートは、これらの相互作用に対して

・協調プロセス群／協調作業モデルの構成／形態／振舞いの記述を可能にし、

・その記述に基づいた、連携のプログラミングおよび制御手段を提供することにより実現することができる。

【０１８２】CCFでは、協調プロセス群の間の相互作用をNodeモデルによって記述／制御する。また、協調作業モデル間のサポートとしては、モデルの構成要素間の関係を記述し、制御するためのプログラマ・インタフェースを提供している。

【０１８３】ただし、本来は協調作業モデル間のサポートにおいて、構成要素間の関係以外の連携も考慮する必要がある。又本発明は前述した協調作業のソフトウェアを担持した媒体であってコンピュータで読み出すことができる媒体も特徴とする。

【０１８４】

【発明の効果】以上の様にして本発明によれば

・さまざまな協調作業形態をモデル化し、それをサポートする協調作業支援システムの作成が容易となる。

・さまざまな協調作業形態における状態の変化をモデル化し、それに対応したシステムの実現が容易となる。

・さまざまな協調作業形態を連係させ、ある協調作業形態から他の協調作業形態への移行を実現することが容易となる。

・協調作業形態の相違点や拡張の影響をうけないアプリケーションの実装が可能となる。

等の効果を奏する。

【図面の簡単な説明】

【図１】本発明の第１の実施例である会議システムの構成図

【図２】第１の実施例におけるシステムの概要を説明するための図

【図３】Ｃ／ＡオブジェクトモデルにおけるＣｏｌｌａｂＦｉｅｌｄを説明する図

【図４】Ｃ／ＡオブジェクトモデルにおけるＡｃｔｉｖｉｔｙを説明する図

【図５】会議システムが形成するＣ／Ａオブジェクトの例を示す図

【図６】本システムの処理の流れを説明するための図

【図７】図６における処理をフローチャートであらわした図

【図８】本発明の第２の実施例である遠隔状況把握システムの構成図

【図９】第２の実施例におけるシステムの概要を説明するための図

【図１０】状況把握システムが形成するＣ／Ａオブジェクトの例を示す図

【図１１】本システムの処理の流れを説明するための図

【図１２】図１１における処理をフローチャートであらわした図

【図１３】本発明の他の実施例の状況把握アプリ２０００からビデオ会議アプリ３０００への移行を示す図

【図１４】他の実施例のＣＳＣＷアプリの連携を示す図

【図１５】他の実施例のＣＣＦ　Ｃｏｒｅの構成を示す図

【図１６】他の実施例の協調プロセス群間の連携を示す図

【図１７】他の実施例の協調プロセス群間のプロセスの移行を示す図

【図１８】他の実施例の通信路の生成／消去を示す図

【図１９】図１５に示すＷｏｒｋ　Ｅｎｖ　レイヤ３６００の連携サポートを示す図

【符号の説明】

１０１　会議システムが動作しているワークステーション

１０２　ワークステーション１０１を構成するＣＰＵ

１０３　同じくワークステーション１０１を構成する主記憶

１０４　ユーザインタフェースを提供する端末

１０５　ＣＰＵ１０２と主記憶１０３を結合する計算機バス

１０６　会議システムが利用する情報を蓄積するハードディスク

１０７　ハードディスクで管理されているＣ／Ａオブジェクト

１０８　ハードディスクで管理され会議をモデル化している協調作業モデル記述オブジェクト

１０９　１０７および１０８のデータの一貫性管理をおこなうＷａｔｃｈｅｒサーバ

１１０　会議システムを制御するＳｅｓｓｉｏｎ　ｍａｎａｇｅｒ

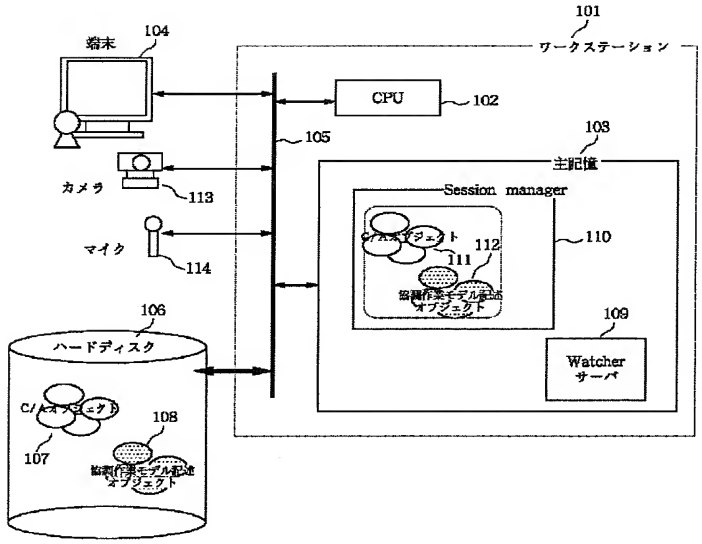
１１１　Ｓｅｓｓｉｏｎ　ｍａｎａｇｅｒ１１０に保持されているＣ／Ａオブジェクト１０７のコピー

１１２　Ｓｅｓｓｉｏｎ　ｍａｎａｇｅｒ１１０に保持されている協調作業モデル記述オブジェクト１０８のコピー

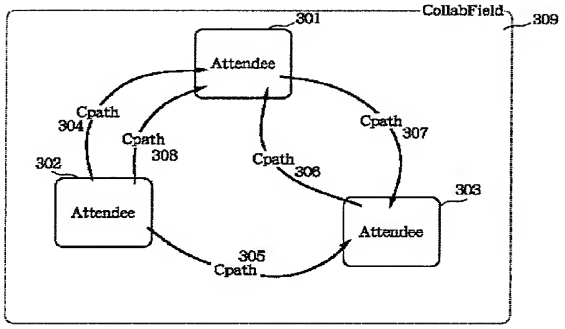
１１３　動画をとりこむためのカメラ

１１４　音声をとりこむためのマイク

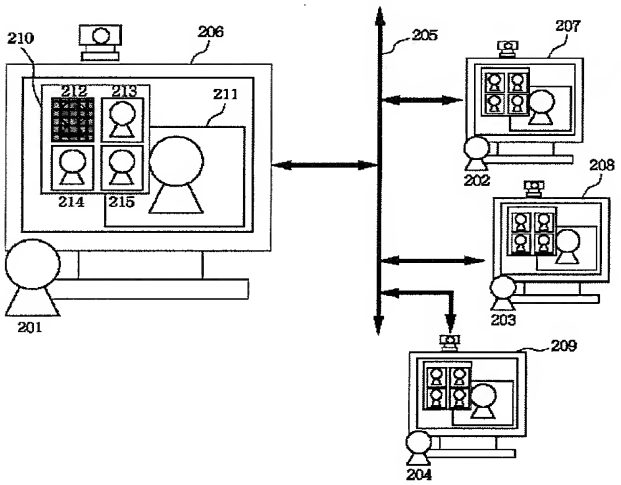
【図１】


101
ワークステーション
端末
104
CPU
102
105
103
主記憶
カメラ
113
Session manager
マイク
114
112
111
110
106
ハードディスク
109
Watcher
サーバ
108
107

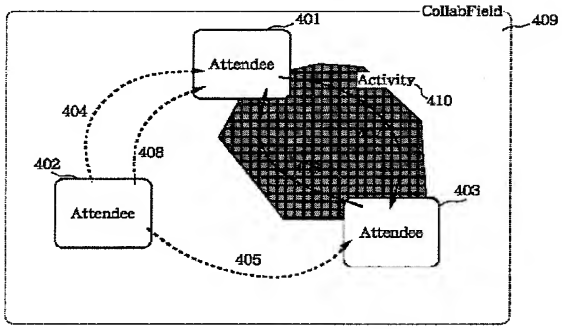
【図３】


CollabField
309
301
Attendee
Cpath
304
Cpath
308
Cpath
307
Cpath
306
302
Attendee
303
Attendee
305
Cpath

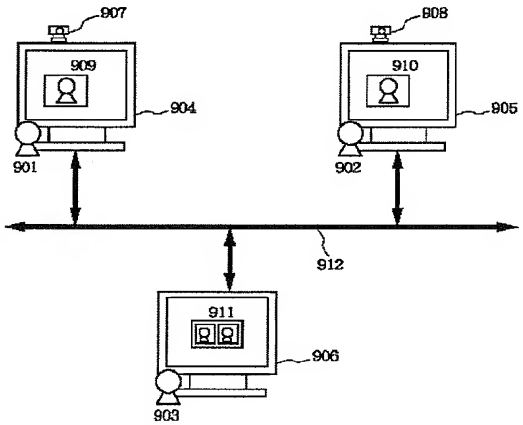
【図２】


210
206
205
207
212
213
211
214
215
202
208
201
203
209
204

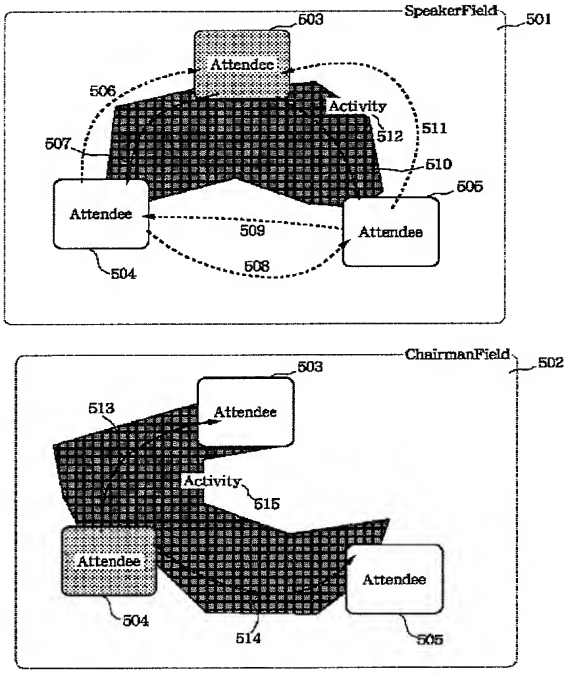
【図４】


CollabField
409
401
Attendee
Activity
410
404
408
402
Attendee
403
Attendee
405

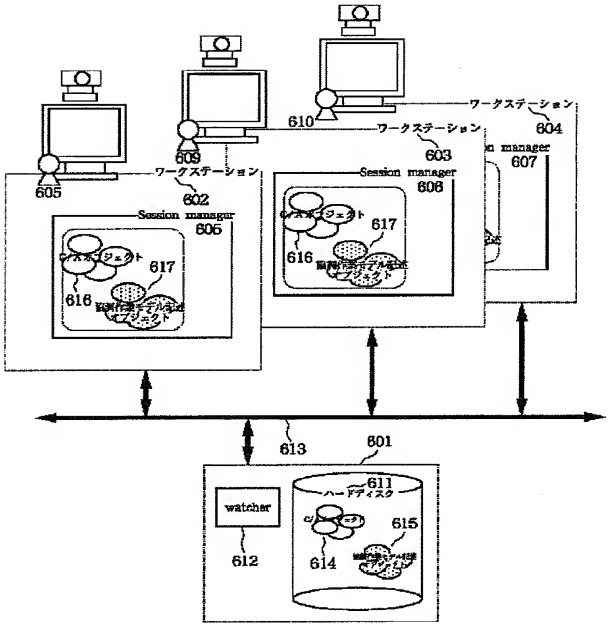
【図９】


907
909
904
901
908
910
905
902
912
911
906
903

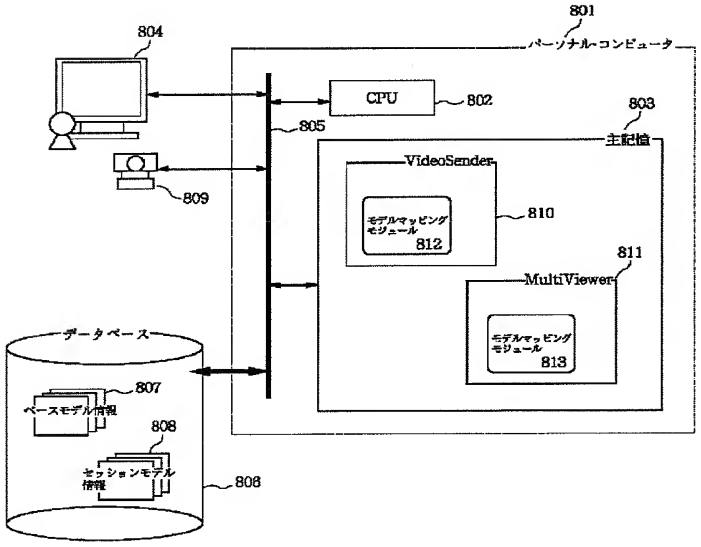
【図５】


SpeakerField
501
503
Attendee
506
Activity
512
511
507
510
505
Attendee
509
Attendee
504
508
ChairmanField
502
503
513
Attendee
Activity
515
Attendee
Attendee
504
514
505

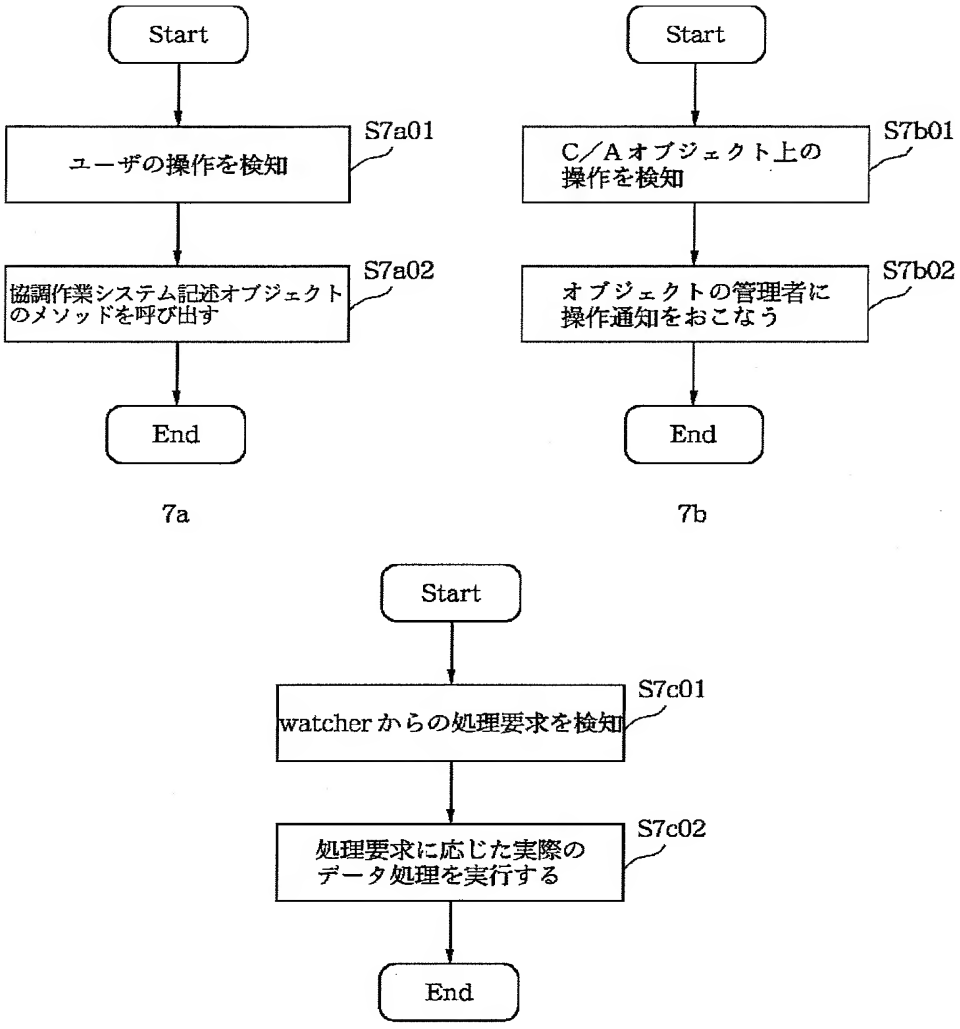
【図６】


ワークステーション
604
610
ワークステーション
603
609
ワークステーション
602
605
Session manager
607
Session manager
606
Session manager
605
617
616
617
616
613
601
611
watcher
615
612
614

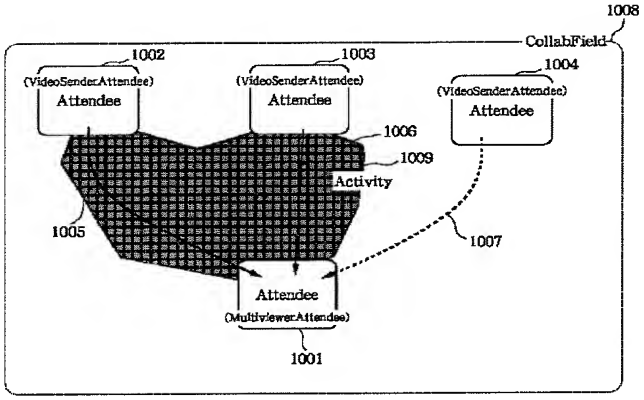
【図８】


801
パーソナルコンピュータ
804
CPU
802
803
主記憶
805
VideoSender
809
モデルマッピング
モジュール
812
810
811
MultiViewer
モデルマッピング
モジュール
813
データベース
807
ベースモデル情報
808
セッションモデル
情報
806

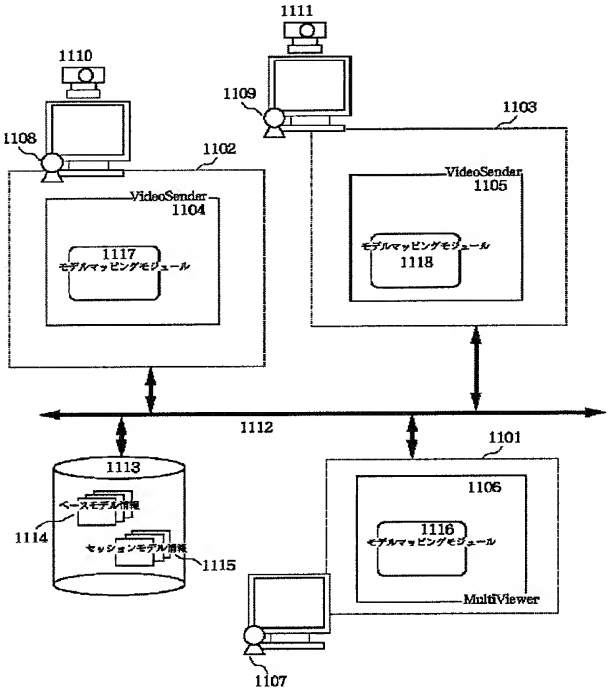
【図７】


Start
ユーザの操作を検知
S7a01
協調作業システム記述オブジェクトのメソッドを呼び出す
S7a02
End
7a
Start
C／Aオブジェクト上の操作を検知
S7b01
オブジェクトの管理者に操作通知をおこなう
S7b02
End
7b
Start
watcherからの処理要求を検知
S7c01
処理要求に応じた実際のデータ処理を実行する
S7c02
End
7c

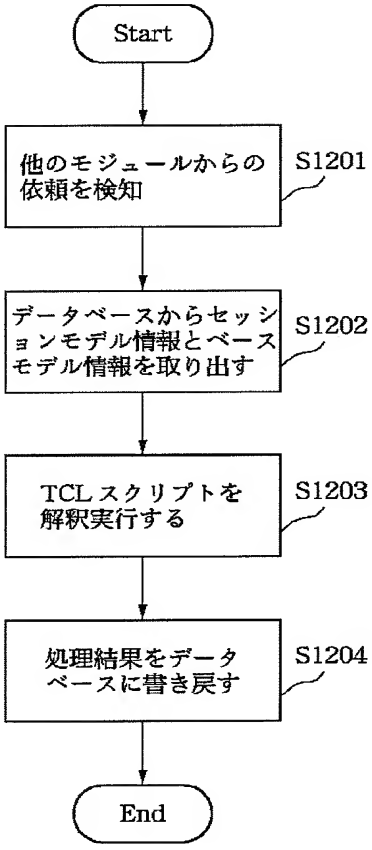
【図１０】



【図１１】

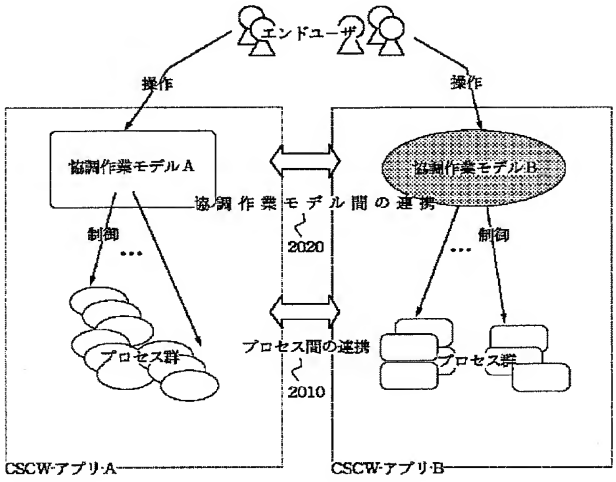


【図１２】



【図１４】

CSCWアプリの連携

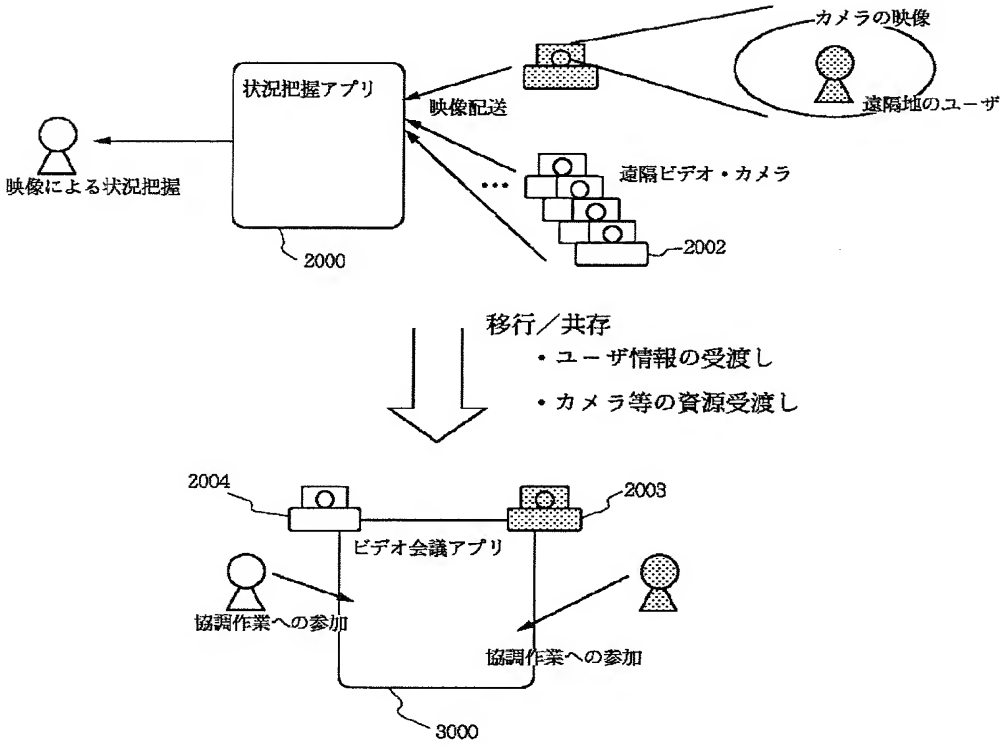


【図１３】

(CSCWアプリの連携)

◆複数のCSCWアプリ間の連携（Example）

・状況把握（Awaraness）からのDesktop会議の開始

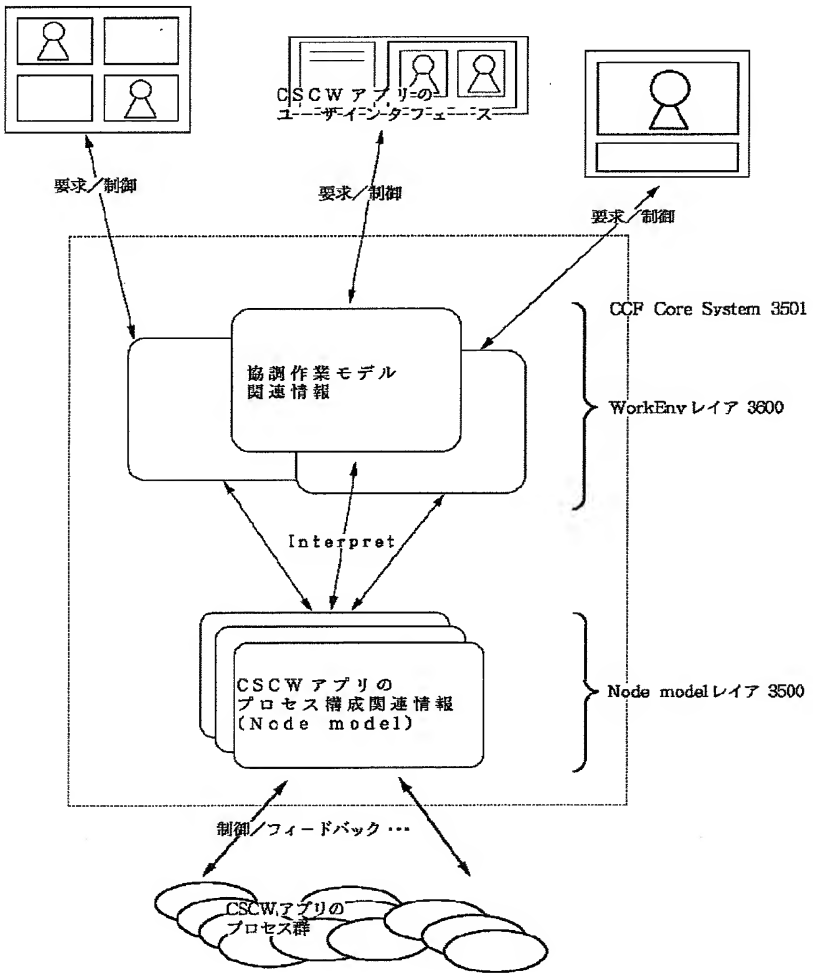


【図１５】

(CCFの概要)

◆CCFの構成
- CCF Toolkit－C＋＋プログラマ・インタフェース
- CCF Core System－CSCWアプリに関する情報を管理するサーバ

◆CCFのレイヤ構造
- Node modelレイヤー協調プロセス群の管理／制御レイヤ
- WorkEnvレイヤー協調作業モデルの管理／制御レイヤ



【図１６】

（協調プロセス群間の連携）

◆協調プロセス群の記述モデル

・制御の対象となるプロセス／資源

・プロセス／資源間の通信路（アクセス）

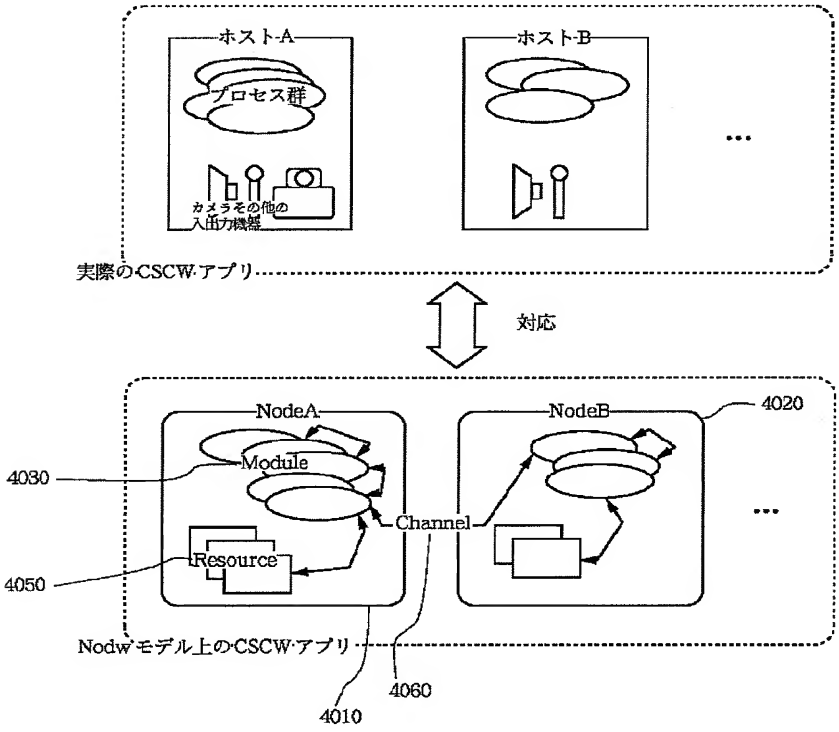
◆Node モデル

CSCW アプリを構成する協調プロセス群を、

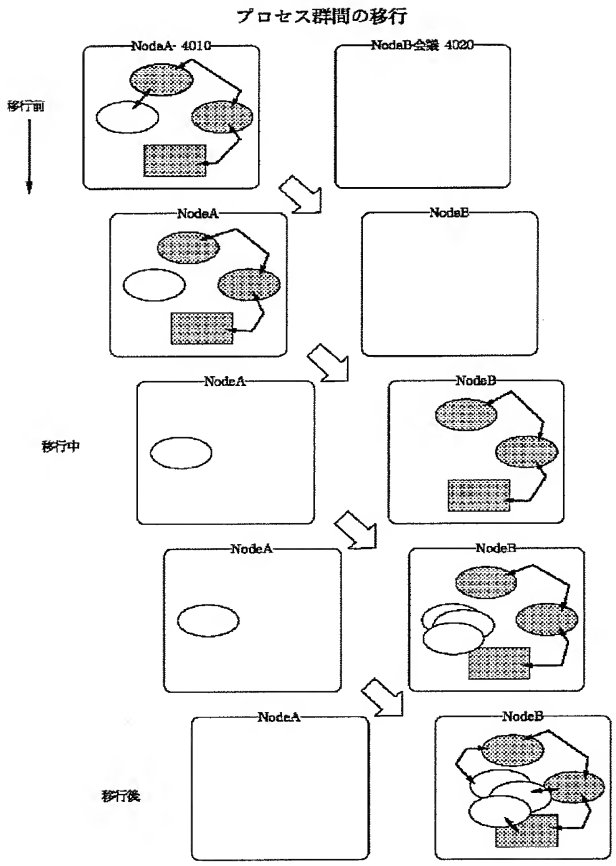
・CSCW アプリが作動するホストやユーザ環境を表現する Node と、

・Node 内で動作／管理されるプロセス／資源（Nodule,Resource,Channel）の集まり

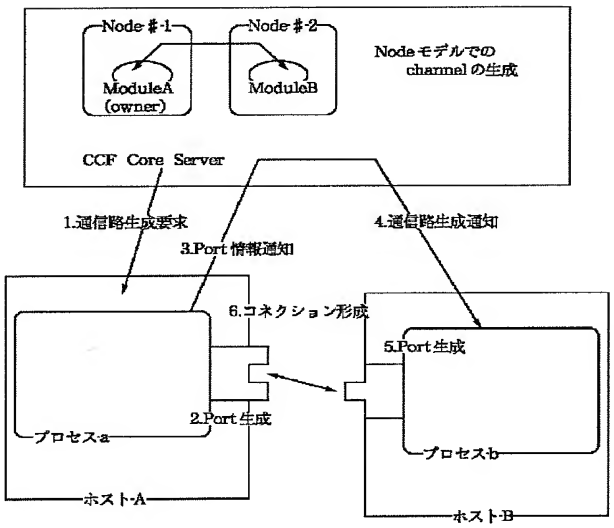
として記述する。



【図１７】


プロセス群間の移行
移行前
NodeA 4010
NodeB会議 4020
NodeA
NodeB
移行中
NodeA
NodeB
NodeA
NodeB
NodeA
NodeB
移行後

【図１８】


(Channelによる通信路の形成)
Node #1
Node #2
ModuleA
(owner)
ModuleB
Nodeモデルでの
channelの生成
CCF Core Server
1.通信路生成要求
3.Port情報通知
4.通信路生成通知
6.コネクション形成
5.Port生成
2.Port生成
プロセスa
プロセスb
ホストA
ホストB

【図１９】

（WorkEnv レイヤでの連携サポート）